
Problem Set 1 Solutions

Solution to **Problem 1: Ill-Logical**

We are asked to check the boolean equality $\overline{P} + (P \cdot V) = V + \overline{P}$. Here we present three ways to test this identity: direct algebraic manipulation, hand writing the truth tables, and a MATLAB calculation.

Algebraic Manipulation

Referencing the manipulations drawn from [Table 1.4, Properties of Boolean Algebra](#) in the course notes, we see that the Unnamed Theorem is probably most applicable to this problem. By creating a dummy variable $C = \overline{P}$ we produce:

$$(V \cdot \overline{C}) + C = C + V \quad (1-3)$$

which we see is exactly the Unnamed theorem. Thus we conclude that the identity is true.

Truth Table Analysis

We can also prove this identity by writing out the truth tables. By hand it looks like this, with all intermediate calculations:

P	V	\overline{P}	$P \cdot V$	$P \cdot V + \overline{P}$	$V + \overline{P}$
0	0	1	0	1	1
0	1	1	0	1	1
1	0	0	0	0	0
1	1	0	1	1	1

Table 1-1: Truth table analysis of Problem 1

MATLAB Program

Using the following code, we can solve the problem in MATLAB:

```
% Start of file, filename testequality.m
%
P = [0 0 1 1];
V = [0 1 0 1];
Right = V | ~P;
Left = (P & V) | (~P);
if Left == Right
    fprintf('The equality is true.\n');
else
```

```

    fprintf('The equality is false.\n');
end

%%% End of file

```

This code gives the following output:

```

>> testequality
The equality is true.

```

Solution to Problem 2: Digital Monotonicity

Solution to Problem 2, part a.

Each variable A and B has two possible values, making four different combinations of inputs (A, B) . Each combination of inputs (four possible) can cause one of two output values. Therefore the number of possible one-output binary functions of two binary variables is 2^4 , or 16. They are enumerated in the table below.

		$F(A, B)$															
A	B	b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}	b_{12}	b_{13}	b_{14}	b_{15}
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Table 1–2: Truth table analysis of Problem 2, part a

There are six possible transitions that involve an increase in either A or B inputs, that is, $[0, 0] \rightarrow [0, 1]$, $[0, 0] \rightarrow [1, 0]$, $[0, 0] \rightarrow [1, 1]$, $[0, 1] \rightarrow [1, 0]$, $[0, 1] \rightarrow [1, 1]$, and $[1, 0] \rightarrow [1, 1]$. By examining each column in the table we can determine which functions are monotonic-increasing. They are *AND* (b_8), *OR* (b_{14}), 1 (b_{16}), 0 (b_0), A (b_{12}), and B (b_{10}).

Solution to Problem 2, part b.

We can easily see that the function is monotonic-increasing once we have written and run the following MATLAB program. Upon reflection, we might realize that all feed-forward cascades of monotonic-increasing functions are themselves monotonic-increasing.

```

% Calculate Problem 1.2 circuit topology

fprintf(1, 'A B C D E -> F\n')

for i = 0:31
    IN = bitget(i,5:-1:1);
    A = IN(1);
    B = IN(2);
    C = IN(3);
    D = IN(4);
    E = IN(5);

```

```
F = bitor(bitor(bitor(A, B), C), bitor(D, E));
fprintf(1, '%d %d %d %d %d %d\n', A, B, C, D, E, F);
end
%
```

This code gives the following output:

```
A B C D E ->F
0 0 0 0 0 0
0 0 0 0 1 1
0 0 0 1 0 1
0 0 0 1 1 1
0 0 1 0 0 1
0 0 1 0 1 1
0 0 1 1 0 1
0 0 1 1 1 1
0 1 0 0 0 1
0 1 0 0 1 1
0 1 0 1 0 1
0 1 0 1 1 1
0 1 1 0 0 1
0 1 1 0 1 1
0 1 1 1 0 1
0 1 1 1 1 1
1 0 0 0 0 1
1 0 0 0 1 1
1 0 0 1 0 1
1 0 0 1 1 1
1 0 1 0 0 1
1 0 1 0 1 1
1 0 1 1 0 1
1 0 1 1 1 1
1 1 0 0 0 1
1 1 0 0 1 1
1 1 0 1 0 1
1 1 0 1 1 1
1 1 1 0 0 1
1 1 1 0 1 1
1 1 1 1 0 1
1 1 1 1 1 1
```